JOURNAL OF COMBINATORIAL THEORY (A) **22**, 17–30 (1977)

Lexicographic Compositions and deBruijn Sequences

HAROLD FREDRICKSEN AND IRVING KESSLER*

IDA, Communications Research Division, Thanet Road, Princeton, New Jersey 08540

Communicated by the Managing Editors

Received April 30, 1975

In this paper we give an algorithm to generate a new deBruijn sequence. The amount of storage required is only linear in n, the span of the sequence. We make use of the lexicographic compositions of a positive integer which we relate to binary necklaces of beads in two colors. We give a fast algorithm for generating the necklaces. Finally we relate the deBruijn sequence we generate to another sequence.

INTRODUCTION

In this paper we give an algorithm to generate a deBruijn sequence of span n. The sequence we generate requires at most an amount of storage linear in n. Our sequence is very close to another sequence generated by Eldert *et al.* [1]. Their sequence requires storage of all 2^n bits of the sequence, however. A proof of the Eldert algorithm appears in [5].

Our algorithm is based upon properties of ordered partitions of the integer $n - 1$ which we call lexicographic compositions. These compositions were studied earlier as cyclic partitions by Motzkin [9]. The compositions are also related to necklaces of beads of length $n - 1$ in two colors. We show the equivalence and give a new, fast algorithm for generating the necklaces.

In Section 1 we define the lexicographic compositions and relate them to the necklaces of beads in two colors.

In Section 2 we give an algorithm (Algorithm A) for generating the lexicographic compositions and state and prove some properties of the list generated which are used to prove our main theorems, Theorems 4 and 5.

In Section 3 we define a deBruijn sequence and we give an algorithm (Algorithm B) for generating our sequence of span n.

In Section 4 we consider the relationship of our sequence to that of Eldert *et al.* [1].

* Current address: Southern Illinois University, Edwardsville, Illinois 62026.

1. Lexicographic Compositions and Necklaces

Definitions

1. Let $a_1, a_2, \ldots, a_k$ be positive integers. We call the k-tuple $a_1 \cdots a_k$ a *composition of* n if $a_1 + \cdots + a_k = n$.

2. If $a_1 \cdots a_k$ and $b_1 \cdots b_t$ are compositions of n, we say $a_1 \cdots a_k \geqslant b_1 \cdots b_t$, if for some j, $a_j \geqslant b_j$ and $a_i = b_i$ for $i < j$.

3. A *lexicographic composition* is a composition $a_1 \cdots a_k$ such that $a_1 \cdots a_k \geqslant a_{\pi(1)} \cdots a_{\pi(k)}$ where π is any cyclic permutation of $\{1, \ldots, k\}$.

4. A composition $a_1 \cdots a_k$ is *periodic* if $a_1 \cdots a_k = a_{\pi(1)} \cdots a_{\pi(k)}$ for some nontrivial cyclic permutation π of $\{1, \ldots, k\}$. If a composition is periodic and we choose the smallest $j > 1$ such that $a_1 \cdots a_k = a_j \cdots a_k a_1 \cdots a_{j-1}$ then we call $a_1 \cdots a_{j-1}$ a *period of length* $j - 1$ of $a_1 \cdots a_k$.

Given a necklace of length n in two colors of beads (i.e., a circular string of n bits), we can associate with that necklace several compositions of n, defined by the set of distances between successive *ones* on the necklace, with the exception of the necklace consisting of all zeroes. Two compositions will represent the same necklace if and only if one composition is a cyclic permutation of the other. Thus, with the exception of the all zero necklace, there is a one-to-one correspondence between the set of necklaces of length n in two colors and the set of lexicographic compositions of n.

The number of such necklaces is well known [7, 10, p. 162], to be

$$Z_n = (1/n) \sum_{d \mid n} \phi(d)\, 2^{n/d},$$

where ϕ is Euler's totient function.

Hence the number of lexicographic compositions of n is $Z_n - 1$.

Consider the 2^n binary strings of length n. We call two strings S_1 and S_2 equivalent if there exists a cyclic permutation of the bits π such that $\pi S_1 = S_2$. Then the necklaces are the equivalence classes. We say a necklace has period p, if the corresponding lexicographic composition has a period and the sum of the parts in the periodic portion is p, otherwise we say the necklace has period n. Then the number of binary strings in an equivalence class determined by a necklace will be equal to the period of the necklace.

Consider a list of lexicographic compositions of n, where *we only write a period* composition if the composition is periodic. Then the sum of the parts in a composition in this list will be the number of binary strings in the equivalence class determined by the corresponding necklace. Thus, the sum of all the parts in this list of compositions will be $2^n - 1$.

2. Generating the Lexicographic Compositions

Notation. Let LC(m) denote the set of lexicographic compositions of m.

We give below an algorithm for generating a lexicographic list of the lexicographic compositions of the integer m. By our remarks above, the algorithm also generates a list of necklaces of length m, of beads in two colors. A more complete analysis of the algorithm appears in [6].

Algorithm A. Let $s_1 = m$ be the first element in the list and let $s_j = a_1 \cdots a_k$ denote the jth element in our list. We construct s_{j+1} as follows:

($*$) Choose the largest i such that $a_i > 1$.

Let $b_1 = a_1$, $b_2 = a_2, \ldots, b_{i-1} = a_{i-1}$, $b_i = a_i - 1$ and for $t > i$, let $t = iq + r$ with $1 \leqslant r \leqslant i$, we then define

$$b_t = b_r, \qquad \text{if} \quad \sum_{j=1}^{t-1} b_j + b_r < m,$$

and

$$b_t = m - \sum_{j=1}^{t-1} b_j, \qquad \text{otherwise.}$$

Let $s'_{j+1} = b_1 \cdots b_\ell$. If $s'_{j+1} \in \text{LC}(m)$, set $s_{j+1} = s'_{j+1}$.

If $s'_{j+1} \notin \text{LC}(m)$, we apply ($*$) to s'_{j+1}. We continue this process until we reach an element of LC(m), which must occur since the process will eventually yield the composition consisting of all ones.

Example. We apply Algorithm A to generate the lexicographic list of LC(6). The arrows represent the result of applying ($*$). The composition which appears that is not lexicographic and hence is rejected by the algorithm appears in parentheses.

$$6 \to 5, 1 \to 4, 2 \to 4, 1, 1 \to 3, 3 \to 3, 2, 1 \to 3, 1, 2$$
$$\to 3, 1, 1, 1 \to 2, 2, 2 \to 2, 2, 1, 1 \to 2, 1, 2, 1$$
$$\to (2, 1, 1, 2) \to 2, 1, 1, 1, 1 \to 1, 1, 1, 1, 1, 1.$$

Theorem 1. *Every lexicographic composition appears in the sequence generated by Algorithm* A.

Proof. Let $s_j = a_1 \cdots a_k$ be the jth element in the sequence generated by Algorithm A. We exhibit the inverse to process ($*$).

$(*)^{-1}$ Choose the smallest i such that for $i \leqslant t \leqslant k$, $t \equiv t' \pmod{i}$ implies $a_t = a_{t'}$ (Note: If we take $i = k - 1$ we will always satisfy the condition.) Let $b_1 = a_1, \ldots, b_{i-1} = a_{i-1}$, $b_i = a_i + 1$, and let $b_r = 1$ for $i < r \leqslant q$, where $q = m - (b_1 + \cdots + b_i)$.

Let $s'_{j-1} = b_1 \cdots b_q$. Then applying $(*)$ to s'_{j-1} yields s_j. Thus, if $s'_{j-1} \in \mathrm{LC}(m)$, we see that $s_{j-1} = s'_{j-1}$. If $s'_{j-1} \notin \mathrm{LC}(m)$ we apply $(*)^{-1}$ to s'_{j-1} repeating the process until we arrive at an element of $\mathrm{LC}(m)$. Since $(*)(*)^{-1}$ is the identity map on compositions, the first lexicographic composition we arrive at must be s_{j-1}.

We complete this section with three lemmas, which give some elementary properties of the sequence generated by Algorithm A. The lemmas are used to prove our main results in Section 3.

LEMMA 1. *Let* $L = a_1 \cdots a_k \in \mathrm{LC}(m)$ *and let* $G = a_1 \cdots a_i b_1 \cdots b_t$ *where* $1 \leqslant i \leqslant k - 1$, $t = m - (a_1 + \cdots + a_i)$, *and* $b_j = 1$ *for* $1 \leqslant j \leqslant t$. *Then* $G \in \mathrm{LC}(m)$.

Proof. If $t = k - i$, then $G = L$ and we are done. So assume $t > k - i$.

Suppose that G', $G' \neq G$, was the lexicographic composition obtained by cyclicly permuting the parts of G, say

$$G' = a_{j+1} \cdots a_i b_1 \cdots b_t a_1 \cdots a_j .$$

Let $J_1 = a_1 \cdots a_j$, $J_2 = a_{j+1} \cdots a_{2j}, \ldots$, and let J_c be the j-tuple which contains b_1. Let J_c' be the corresponding j-tuple from L, viz,

$$J_c' = a_{(c-1)j+1}, \ldots, a_k \qquad \text{if } k \leqslant cj,$$

or

$$J_c' = a_{(c-1)j+1}, \ldots, a_{cj} \qquad \text{if } k > cj .$$

Now, $J_1 \geqslant J_i$ for $1 \leqslant i \leqslant c - 1$, since $L \in \mathrm{LC}(m)$. But $J_2 \geqslant J_1$, since $G' \in \mathrm{LC}(m)$, so $J_2 = J_1$. Now $G' \geqslant G$ and $J_1 = J_2$ implies $J_3 \geqslant J_2$. So $J_3 \geqslant J_2 = J_1 \geqslant J_3$, thus $J_1 = J_2 = J_3$. Similarly we can obtain $J_1 = \cdots = J_{c-1} = J_c$.

J_c must contain the entire composition $b_1 \cdots b_t$, for if not, a continuation of the above argument gives $J_1 = J_{c+1} = b_r \cdots$ for some r, which implies $a_1 = b_r = 1$, a contradiction. So $J_c' > J_c$, since $t > k - i$. Hence $J_1 \geqslant J_c' > J_c = J_1$, a contradiction showing $G \in \mathrm{LC}(m)$.

LEMMA 2. *Suppose* S, T, *and* $U \in \mathrm{LC}(m)$, *where* $S \geqslant T \geqslant U$, *and*

$$S = a_1 \cdots a_k ,$$
$$T = b_1 \cdots b_h ,$$
$$U = c_1 \cdots c_j .$$

If $c_1 = a_1, \ldots, c_i = a_i$, *then* $b_1 = a_1, \ldots, b_i = a_i$.

Proof. $T \geqslant U$ implies that there exists a t such that $b_1 = c_1, \ldots, b_{t-1} = c_{t-1}$, and $b_t > c_t$. If $t > i$, we are done. If $t \leqslant i$, then $S \geqslant T$ implies that there exists a v such that $a_1 = b_1, \ldots, a_{v-1} = b_{v-1}$, and $a_v > b_v$. If $v > t$, $a_t = b_t > c_t$, a contradiction. If $v \leqslant t$, $a_v > b_v = c_v$, a contradiction. Thus $t > i$ and the lemma is proven.

LEMMA 3. *Suppose* $s_j = a_1 \cdots a_k$ *and* $s_{j+1} = c_1 \cdots c_h$ *are consecutive lexicographic compositions generated by Algorithm* A, *with* $a_i > 1$ *and* $a_j = 1$ *for* $i < j \leqslant k$. *Then*

$$a_1 = c_1, \ldots, a_{i-1} = c_{i-1}, \qquad c_i \geqslant a_i - 1.$$

Proof. Applying Algorithm A to s_j yields $s'_{j+1} = b_1 \cdots b_i \cdots b_1 \cdots b_i x$ where $b_1 = a_1, \ldots, b_{i-1} = a_{i-1}$, $b_i = a_i - 1$, and x is some (perhaps empty) string. If $s'_{j+1} \in \mathrm{LC}(m)$ we are through. If $s'_{j+1} \notin \mathrm{LC}(m)$, we apply Algorithm A to s'_{j+1} repeatedly until we reach a lexicographic composition.

Let $T = b_1 \cdots b_i 1 \cdots 1$. Then $T \in \mathrm{LC}(m)$ by Lemma 1 and $s_j \geqslant s_{j+1} \geqslant T$. Hence by Lemma 2, $a_1 = c_1, \ldots, a_{i-1} = c_{i-1}$, and the definition of the order among compositions gives $c_i \geqslant b_i = a_i - 1$.

3. A deBRUIJN SEQUENCE

Definitions

1. Let V_n denote the n dimensional vector space over the field $GF(2)$. A *binary n-tuple* is a vector in V_n.

2. A *deBruijn sequence of order n* or (deBruijn cycle of order n) is a sequence of 2^n elements from $GF(2)$ such that every element of V_n occurs precisely once in the sequence. The end of the sequence is considered to be contiguous with the beginning of the sequence.

deBruijn sequences have been constructed by Martin [8] and Eldert *et al.* [1]. These algorithms used information from the beginning of the sequence throughout the construction, hence, 2^n bits of storage is required for their generation. The first algorithm was refined by Fredricksen [2–4], so that an amount of storage only linear in n was required to produce the sequence. In this section we produce an algorithm for the construction of another deBruijn sequence of span n which requires an amount of storage linear in n, and is closely related to the sequence of Eldert *et al.* [1]. In the final section we discuss the relationship of our sequence to the sequence of Eldert.

Notations and Conventions

Let $s_j = a_1 \cdots a_k \in \mathrm{LC}(m)$.

1. The symbol $[a_1 \cdots a_k]$ denotes a string of a_1 ones, a_2 zeroes, a_3 ones, etc., and the symbol $(a_1 \cdots a_k)$ denotes a string of a_1 zeroes, a_2 ones, a_3 zeroes etc. The symbol $\{s_j\}$ denotes either $[s_j]$ or (s_j).

2. If s_{j+1} is the lexicographic composition following s_j in Algorithm A we let $fs_j = fa_1 \cdots a_k = s_{j+1}$ and $f^t s_j = f^t a_1 \cdots a_k = s_{j+t}$.

3. If s_j is periodic with period p, and the period is repeated t times in s_j, we let $\{a_1 \cdots a_p\}^t = \{a_1 \cdots a_k\}$.

4. $\Lambda s_j = k$, will denote the length of s_j, that is, the number of a_i's.

5. Let $\{a_1 \cdots a_k\} = \{a_1 \cdots a_p\}^t$. For the rest of this paper whenever we say write $\{a_1 \cdots a_k\}$, we will always mean *write* $\{a_1 \cdots a_p\}$. In addition, any reference to the *written* string $\{a_1 \cdots a_k\}$ will be a reference to $\{a_1 \cdots a_p\}$. For example, the length of the written string $\{a_1 \cdots a_k\}$ is p, and we will use the symbol $\Lambda W\{a_1 \cdots a_k\}$ to denote the length of the written string $\{a_1 \cdots a_k\}$.

6. Let $\mathcal{O}(n-1) = \{A \in \mathrm{LC}(n-1)$: A is periodic and $\Lambda W(A)$ is odd$\}$. We will sometimes write $\mathcal{O} = \mathcal{O}(n-1)$ and $\mathrm{LC} = \mathrm{LC}(n-1)$, if the size of the composition is clear from the context.

ALGORITHM B. A method of generating a deBruijn sequence of order n from $\mathrm{LC}(n-1)$.

Step I. Write n ones followed by n zeroes.

Step II. Discard the first lexicographic composition from the list generated by Algorithm A, i.e., discard $n-1$. Write $[n-2, 1]$.

Step III. The inductive step.

1. If we have just written $[a_1 \cdots a_k]$ and $\Lambda W[a_1 \cdots a_k]$ is even, then write $[fa_1 \cdots a_k]$.

2. If we have just written $[a_1 \cdots a_k]$ and $\Lambda W[a_1 \cdots a_k]$ is odd, then either (i) $a_k > 1$ or (ii) for some $j \leqslant k-1$, $a_j > 1$ and $a_t = 1$ for $j < t \leqslant k$. If (i), then write $(a_1 \cdots a_k)$. If (ii), then construct the following compositions of $n-1$.

$$B = b_1 \cdots b_{k-1},$$

where

$$\begin{aligned} b_i &= a_i \qquad \text{for } 1 \leqslant i < j, \\ b_j &= a_j + 1, \\ b_i &= 1 \qquad \text{for } j < i < k, \end{aligned}$$

and

$$C_t = c_1 \cdots c_{k-1},$$

where

$$\begin{aligned} c_i &= a_i \qquad \text{for } 1 \leqslant i \leqslant j, \\ c_i &= 1 \qquad \text{for } j < i < (j+t) \text{ and for } (j+t) < i < k, \\ c_{j+t} &= 2. \end{aligned}$$

Now, if $B \in \mathrm{LC}(n-1)$ and $B \notin \mathcal{O}(n-1)$, then write (B). If $B \notin \mathrm{LC}$ or $B \in \mathcal{O}$, then choose the smallest t such that $C_t \in \mathrm{LC}$ and $C_t \notin \mathcal{O}$, and write (C_t). If no such t exists, then write $(a_1 \cdots a_k)$. Exception: If n is even ($n-1$ is odd), after writing [1] we write (2, 1,..., 1).

3. If we have just written $(a_1 \cdots a_k)$ and $\Lambda W(a_1 \cdots a_k)$ is odd, then write $[fa_1 \cdots a_k]$. Exception: If $a_i = 1$ for all i, then we stop.

4. If we have just written $(a_1 \cdots a_k)$ and $\Lambda W(a_1 \cdots a_k)$ is even, then either (i) $a_k > 1$ or (ii) $a_j > 1$ for some $j \leqslant k-1$, and $a_i = 1$ for $j < i \leqslant k$.

If (i), then let $a_k^- = a_k - 1$ and write $(a_1 \cdots a_{k-1} a_k^- 1)$.

If (ii), then construct the following composition of $n-1$.

$$B = b_1 \cdots b_k$$

where

$$\begin{aligned} b_i &= a_i \qquad \text{for } 1 \leqslant i < j, \\ b_j &= a_j - 1, \qquad b_{j+1} = 2, \end{aligned}$$

and

$$b_i = 1 \qquad \text{for } (j+2) \leqslant i \leqslant k.$$

If $B \in \mathrm{LC}$ and $B \notin \mathcal{O}$, then write (B). If $B \notin \mathrm{LC}$ or $B \in \mathcal{O}$, then write $(b_1 \cdots b_j, 1,..., 1)$.

EXAMPLE. We use Algorithm B to generate a deBruijn sequence of length 2^7 from the lexicographic list of LC(6) (cf. the example following Algorithm A). 1111111/0000000/111110, by Steps I and II.

In the following the numerals 1–4 refer to the parts of Step III we are using. 111100/111101, by applying 1 to [5, 1] and [4, 2]. 000001, by applying 2(ii) to [4, 1, 1]. 000011/000010, by applying 4 to (5, 1) and (4, 2). 111, by applying 3 to (4, 1, 1) and noting that since [3, 3] is periodic, we only write [3]. 000, since ΛW[3, 3] is odd, we apply 2(i). 111001, by applying 3 to (3, 3). 000110, by applying 2(ii) to [3, 2, 1]. 111011/000100/

111010, by 3, 2(i), and 3. 11/00/110010/110/110101, by 1, 2(i), 3, and two applications of 1. 000101/001101/001/001010, by 2(ii), and three applications of 4(ii). 1/0, by 3 and 2(i).

We now stop by 3.

Thus the complete deBruijn sequence of order 7 is

1111111000000011111011110011110100000100001100000 10

1110001110010001101110110001001110101100110010110 1

10101000101001101001001010 10.

Notation. If G and $H \in \mathrm{LC}(n-1)$, then we write $\{G\}/\{H\}$ or $F\{G\} = \{H\}$ to mean $\{H\}$ follows $\{G\}$ in Algorithm B. Let $F^1 = F$ and for $t \geqslant 2$ let $F^t = F \circ F^{t-1}$.

We now will prove that Algorithm B actually generates a deBruijn sequence. The next two theorems show that every lexicographic composition occurs exactly twice in the sequence generated by Algorithm B. Theorem 2 gives the inverse algorithm to Algorithm B.

THEOREM 2. *Suppose we are given* $H \in \mathrm{LC}(n-1)$ *and a choice of brackets* $\{\ \} = (\ )$ or $\{\ \} = [\ ]$. *Then there exists a unique* $G \in \mathrm{LC}(n-1)$ *and a unique choice of brackets such that* $\{G\}/\{H\}$.

Proof. In the proof of this theorem, we will use the phrase "from *Bn*" to mean that the argument follows from Algorithm B, Section III *n*. Also, if a is an integer, we let $a^+ = a + 1$ and $a^- = a - 1$.

1. Suppose $[H]$ is given. From Theorem 1, there exists a unique G such that $fG = H$. That is, H follows G in Algorithm A. If $\Lambda W(G)$ is even, then from Algorithm B 1, $[G]/[H]$. If $W(G)$ is odd, then from Algorithm B 3, $(G)/[H]$.

2. Suppose we are given $(H) = (a_1 \cdots a_k)$, with $\Lambda W(H)$ odd.

(i) If $a_k > 1$, then $[H]/(H)$, from B 2.

(ii) If $a_i = 1$ for $1 \leqslant i \leqslant k$, then for $n-1$ even, $[1]/(1)$ from B 2, and for $n-1$ odd $(2, 1, \ldots, 1)/(1)$ from B 4.

(iii) Otherwise find j such that $a_j > 1$ and $a_i = 1$ for $j < i \leqslant k$.

If $j = k-1$ and $(a_1 \cdots a_{k-2}a_{k-1}^+) \in \mathrm{LC}$ then $(a_1 \cdots a_{k-2}a_{k-1}^+)/(H)$ from B4. If $(a_1 \cdots a_{k-2}a_{k-1}^+) \notin \mathrm{LC}$, then $[H]/(H)$ from B2. Now if $j \leqslant (k-2)$, construct the compositions $C_t = c_1 \cdots c_{k-1}$, where $c_i = a_i$ for $1 \leqslant i \leqslant j$, $c_{j+t} = 2$, and $c_i = 1$ for $j < i < k$, $i \neq j + t$. Find the largest t such that $C_t \in \mathrm{LC}$ and $C_t \notin \mathcal{C}$. If such a t exists, then $(C_t)/(H)$, from B4. If no such t exists, then $[H]/(H)$, from B2.

3. Suppose we are given $(H) = (a_1 \cdots a_k)$ with $\Lambda W(H)$ even. We find j such that $a_j > 1$ and $a_i = 1$ for $j < i \leqslant k$.

(i) If $a_j > 2$, then $[a_1 \cdots a_j - 1 \cdots 1]/(H)$, from B2.

(ii) If $a_j = 2$ and $a_{j-1} \geqslant 2$ let $G = a_1 \cdots a_{j-2}a_{j-1}^+ 1,..., 1$. If $G \in$ LC and $G \notin \mathcal{O}$ then $(G)/(H)$, from B4. If $G \notin$ LC or $G \in \mathcal{O}$, then $[a_1 \cdots a_{j-1} 1 \cdots 1]/(H)$, from B2. (Note: We have assumed $j \geqslant 2$. If $j = 1$, then $[1]/(2, 1 \cdots 1)$, from B2.)

(iii) If $a_j = 2$ and $a_{j-1} = 1$, then let $G = a_1 \cdots a_{j-2}2, 1 \cdots 1$. If $G \in$ LC and $G \notin \mathcal{O}$, then $(G)/(H)$, from B4. If $G \notin$ LC or $G \in \mathcal{O}$, then choose the largest q such that $q < j$ and $a_q > 1$. If we let $G_1 = a_1 \cdots a_q 1 \cdots 1$, then $[G_1]/(H)$, from B2.

COROLLARY 1. *Let G and $H \in$ LC$(n-1)$, then*

1. *if $F\{G\} = [H]$, then $fG = H$;*
2. *if $F(G) = [H]$, then $\Lambda W(G)$ is odd;*
3. *if $F\{G\} = (H)$ and $\Lambda W(H)$ is odd, then either $\{G\} = [H]$ or $fG = H$.*

Proof. The proof of the corollary is obtained by examining the proof of Theorem 2.

THEOREM 3. *Given n, $G \in$ LC$(n-1)$, and a choice of brackets $\{\ \} = [\ ]$ or $\{\ \} = (\ )$, then there exists a unique t such that $F^t[n-2, 1] = \{G\}$.*

Proof. If a t exists, its uniqueness follows from Theorem 2. Let

$$\mathcal{S}_0 = \{\{L\} : L \in \text{LC}(n-1) \text{ and } F^t[n-2, 1] \neq \{L\}\ \forall t\}.$$

We wish to prove $\mathcal{S}_0 = \varnothing$ so assume $\mathcal{S}_0 \neq \varnothing$ and choose the first H_0 such that $\{H_0\} \in \mathcal{S}_0$ for some choice of the brackets. The ordering throughout this proof is the lexicographic ordering in LC$(n-1)$. For notational convenience we write H for H_0.

If $[H] \in \mathcal{S}_0$, then from Theorem 2 there exists a G such that (i) $G > H$ and (ii) $F(G) = [H]$ or $F[G] = [H]$. From our assumption on H, (i) implies $G \notin \mathcal{S}_0$ and (ii) implies $[H] \notin \mathcal{S}_0$, a contradiction.

If $(H) \in \mathcal{S}_0$ and $\Lambda W(H)$ is odd, then either (i) $F[H] = (H)$ or (ii) there exists $G > H$ such that $F(G) = (H)$. If (i), since $[H] \notin \mathcal{S}_0$, we have $(H) \notin \mathcal{S}_0$, and (ii) contradicts our assumption on H.

Thus we must have $(H) \in \mathcal{S}_0$ with $\Lambda W(H)$ even. From Theorem 2, there exists a G_0 such that $F[G_0] = (H)$, $\Lambda W[G_0]$ is odd, $G < H$, and $G_0 \in \mathcal{S}_0$. Let

$$\mathcal{S}_1 = \{\{Q\} \in \mathcal{S}_0 : G_0 < Q < H\}.$$

If $\mathscr{S}_1 = \varnothing$ and we choose G' such that $fG' = G_0$, then $G' \notin \mathscr{S}_1$ and since $AW[G_0]$ is odd, either $F(G') = [G_0]$ or $F[G'] = [G_0]$. Since $G' \notin \mathscr{S}_1$ and $G' \leqslant H$, $G' \notin \mathscr{S}_0$. Thus there exists a t such that $F^t[n-2, 1] = \{G'\}$, and hence $F^{t+2}[n-2, 1] = (H)$, a contradiction to the assumption on H.

So assume $\mathscr{S}_1 \neq \varnothing$ and choose the largest H_1 such that $\{H_1\} \in \mathscr{S}_1$ for some choice of the brackets $\{\ \}$. If $[H_1] \in \mathscr{S}_1$, then by Theorem 2 and Corollary 1 there exists a G such that $H \geqslant G \geqslant H_1$ and $F\{G\} = [H_1]$. Now $G \notin \mathscr{S}_0$ implies $H_1 \notin \mathscr{S}_0$, so $G \in \mathscr{S}_0$ and hence $G \geqslant H$. Thus $G = H$ and since $[H] \notin \mathscr{S}_0$, we have $F(H) = [H_1]$, which implies $AW(H)$ is odd from Corollary 1, a contradiction. Similarly, if $(H_1) \in \mathscr{S}_1$ and $AW(H_1)$ is odd, then either $F[H_1] = (H_1)$, a contradiction since $[H_1] \notin \mathscr{S}_1$, or there exists a G with $H \geqslant G \geqslant H_1$ such that $F(G) = (H_1)$. As above, we must have $G = H$ and we obtain a contradiction. Thus if $(H_1) \in \mathscr{S}_1$, $AW(H_1)$ is even and there exists a $G_1 < H_1$ with $W[G_1]$ odd and $F[G_1] = (H_1)$, $G_1 \in \mathscr{S}_1$. Let

$$\mathscr{S}_2 = \{\{Q\} \in \mathscr{S}_1 : G_1 < Q < H_1\}.$$

If $\mathscr{S}_2 = \varnothing$, then $\mathscr{S}_1 = \varnothing$ as before. If $\mathscr{S}_2 \neq \varnothing$ we repeat the above construction and we produce a chain

$$\mathscr{S}_0 \supset \mathscr{S}_1 \supset \mathscr{S}_2 \supset \cdots$$

and eventually there exists an n such that $\mathscr{S}_n = \varnothing$, implying $\mathscr{S}_0 = \varnothing$.

COROLLARY 2. *There are* 2^n *bits in the sequence generated by Algorithm* B.

Proof. The sum of all the parts in the list generated from LC($n-1$) by Algorithm A is $2^{n-1} - 1$. In Algorithm B we excluded the single composition $n-1$, added $2n$ bits at the start of the sequence, and Theorem 3 shows that Algorithm B used every other composition in LC($n-1$) precisely twice. Thus there are

$$2(2^{n-1} - 1 - (n-1) + n) = 2^n$$

bits in the sequence generated by Algorithm B.

We now need to show that no duplications occur in the sequence of length 2^n generated by Algorithm B. We require the following two lemmas.

LEMMA 4. *If* $G = a_1 \cdots a_k \in$ LC, *with* $a_j > 1$ *and* $a_i = 1$ *for* $i > j$, *and* $H = b_1 \cdots b_h$, *with* $F\{G\} = \{H\}$, *then*

$$a_1 = b_1, \ldots, a_{j-1} = b_{j-1}, \text{ and } b_j \geqslant a_j - 1.$$

Proof. We consider the four cases given in Step III of Algorithm B.

1. If $F[G] = \{H\}$ and $\Lambda W[G]$ is even, then $H = fG$ and the result follows from Lemma 3.

2. If $F[G] = \{H\}$ and $\Lambda W[G]$ is odd, there are three possibilities for $\{H\}$, all of which satisfy the conclusion of the lemma.

3. If $F(G) = \{H\}$ and $\Lambda W(G)$ is odd, then $H = fG$ and the conclusion follows from Lemma 3.

4. If $F(G) = \{H\}$ and $\Lambda W(G)$ is even, then let $T = a_1 \cdots a_{j-1}\, 1 \cdots 1$. Then $\Lambda W(T) \geqslant \Lambda W(G)$ and $G \geqslant H \geqslant T$. So by Lemma 2 the lemma follows.

LEMMA 5. *If* $G = \{a_1 \cdots a_p\}^t$ *with* $a_1 > 1$ *and* $F\{G\} = \{H\}$, *then* $H = \{a_1 \cdots a_p\}^{t-1} x$, x *is some nonempty string.*

Proof. Choose the largest j, $1 \leqslant j \leqslant p$ such that $a_j > 1$. From Lemma 4, the first $p(t-1) + j - 1$ parts in H agree with those parts in G.

THEOREM 4. *Every binary n-tuple appears in the sequence generated by Algorithm* B.

Proof. Let $e_1 \cdots e_k$ be any composition of n. Then $e_1 \cdots e_k$ will represent a binary n-tuple, by letting e_1 represent a string of e_1 ones (or zeroes), e_2 represent a string of e_2 zeroes (ones), e_3 represent a string of e_3 ones (zeroes), etc.

We will use the notation "———" to show where any given n-tuple can be found among "consecutive" lexicographic compositions as generated by Algorithm B.

Case 1. $e_i = 1$ for $1 \leqslant i \leqslant (k-1)$.

If n is even and e_k represents ones then $(2, 1 \cdots 1)/(1)$[start], where [start] represents the initial string of n ones. If n is odd and e_k represents ones then $(2, 1 \cdots 1)/[1]/(1)/$[start]. If k is even and e_k represents zeroes, then $[e_k 1 \cdots 1]/(e_k{+}1 \cdots 1)$. If k is odd and e_k represents zeroes, then $[e_k{-}1 \cdots 1]/(e_k 1 \cdots 1)$.

Case 2. Let $a = e_1 + e_k - 1$ and suppose $ae_2 \cdots e_{k-1} \in \mathrm{LC}(n-1)$.

(i) Suppose $e_i = 1$ for $2 \leqslant i \leqslant (k-1)$. If e_1 represents ones and $k-1$ is odd, then $[a1 \cdots 1]/(a^+1 \cdots)$, since $e_k < a^+$ and $e_1 \leqslant a$. If e_1 represents ones and $k-1$ is even, then $[a1 \cdots 1]/[a^- \cdots]$, since we are not in Case 1, $a^- = a - 1 \geqslant e_k$. If e_1 represents zeroes and $k-1$ is odd, then $(a1 \cdots 1)/[a^- \cdots]$, with the same argument as above. If e_1 represnts

zeroes and $k-1$ is even, then $(a1 \cdots 1)/(a^{-}2, 1 \cdots 1)$ except if $a = 2$, and then $(2, 1 \cdots 1)/[1]/(1)$.

(ii) For some i, $2 \leqslant i < k$, $a_i > 1$.

If e_1 represents ones, then $[ae_2 \cdots e_{k-1}]/\{b_1 \cdots\}$. For if we find j such that $e_j > 1$ and $e_t = 1$ for $j < t < k$, we have $j > 1$, and by Lemma 4 $b_1 = a \geqslant e_k$. If e_1 represents zeroes, then $(ae_2 \cdots e_{k-1})/\{a \cdots\}$, by the same argument as above.

If $ae_2 \cdots e_{k-1}$ is periodic, say

$$\{ae_2 \cdots e_{k-1}\} = \{a_1 \cdots a_p\}^t.$$

Then $\{a_1 \cdots a_p\}/\{\{a_1 \cdots a_p\}^{t-1} b \cdots\}$ by Lemma 5, and the proof proceeds *mutatis mutandis*.

Case 3. Let $a = e_k + e_1 - 1$ and $H = e_i e_{i+1} \cdots e_{k-1} a e_2 \cdots e_{k-1}$, where $i \geqslant 2$, $H \in \mathrm{LC}(n-1)$, and there exists a j, $1 \leqslant j < i$ such that $e_j > 1$ and $e_t = 1$ for $j < t < i$.

If $j \geqslant 2$, then by Lemma 4, $\{e_i e_{i+1} \cdots e_{k-1} a e_2 \cdots e_j 1 \cdots 1\}/\{e_i \cdots e_{k-1} a \cdots\}$. If $j = 1$, then $a > 1$ and $\{e_i \cdots e_{k-1} a 1 \cdots 1\}/\{e_i \cdots e_{k-1} b \cdots\}$, since by Lemma 4 $b \geqslant a - 1 \geqslant e_k$.

If H is periodic, then we apply Lemma 5 and proceed as in Case 3.

Case 4. Let $a = e_k + e_1 - 1$ and $H = e_i \cdots e_{k-1} a e_2 \cdots e_{i-1}$, where $H \in \mathrm{LC}(n-1)$ and $e_t = 1$ for $1 \leqslant t \leqslant (i-1)$.

If $i = 2$, we have $\{\cdots b\}/\{e_2 \cdots e_k 1\}$. So assume $i \geqslant 3$ and note that $H = e_i \cdots e_k 1 \cdots 1$.

(i) e_i represents zeroes and $\Lambda W(H)$ is even, choose j, $i \leqslant j \leqslant k$, such that $e_j > 1$ and $e_t = 1$ for $j < t \leqslant k$. If $e_j > 2$, then $[e_i \cdots e_j^{-} 1 \cdots 1]/(e_i \cdots e_k 1 \cdots 1)$.

If $e_j = 2$ and $e_{j-1} \geqslant 2$, then let $G = e_i \cdots e_{j-2} e_{j-1}^{+} 1 \cdots 1$. If $G \in \mathrm{LC}(n-1)$ and $G \notin \mathscr{C}$, then $(G)/(H)$ and $(e_i \cdots e_{j-2} e_{j-1}^{+} 1 \cdots 1)/(e_i \cdots e_k 1 \cdots 1)$. If $G \notin \mathrm{LC}$ or $G \in \mathscr{C}$, then $[e_i \cdots e_{j-1} 1 \cdots 1]/(e_i \cdots e_k 1 \cdots 1)$.

If $e_j = 2$ and $e_{j-1} = 1$, then let $G = e_i \cdots e_{j-2} 2, 1 \cdots 1$. If $G \in \mathrm{LC}(n-1)$ and $G \notin \mathscr{C}$ then $(e_i \cdots e_{j-2} 2, 1 \cdots 1)/(e_i \cdots e_k 1 \cdots 1)$. If $G \notin \mathrm{LC}$ or $G \in \mathscr{C}$, then choose q, $i < q < j$ such that $e_q > 1$ and $e_t = 1$ for $q < t < j$. Then $[e_i \cdots e_q 1 \cdots 1]/(e_i \cdots e_k 1 \cdots 1)$.

(ii) e_i represents zeroes and $\Lambda W(H)$ is odd.

Let $G = e_i \cdots e_{k-1} b \cdots$ be the first element of $\mathrm{LC}(n-1)$ with the property that $\Lambda W(G)$ is even and G begins with the string $e_i \cdots e_{k-1}$. If $b \geqslant e_k$ and

$b > 1$, then $[e_i \cdots e_{k-1}b\overline{-1 \cdots 1}]/(e_i \cdots e_{k-1}b \cdots)$. If $b = e_k = 1$, then we find the largest j such that $e_j > 1$ and we have $[e_i \cdots e_j{}^{-}1 \cdots 1]/(e_i \cdots e_{k-1}b \cdots)$. If $b < e_k$, then find the first element of LC$(n-1)$ which begins with the string $e_i \cdots e_{k-1}$, say it is $G = e_i \cdots e_{k-1}b_1 \cdots b_s$. Then we must have $\Lambda W(G)$ is odd, $s = i - 1$, $b_1 = e_k$, $b_j = 1$ for $2 \leqslant j \leqslant (i-1)$, and $F[G] = (G)$. Since $H \in$ LC and begins with the string $e_i \cdots e_{k-1}$, we have $b_1 \geqslant e_k$. If $b_1 > e_k$, then $(e_i \cdots e_{k-1}b_1{}^{-}1 \cdots 1)$ has even length, contradicting the fact that we could not find an even length element of LC which begins with the string $e_i \cdots e_{k-1}b$ with $b > e_k$. And if $b_s > 1$ for some $s \geqslant 2$, then $\Lambda W(e_i \cdots e_{k-1}b_1 \cdots b_{s-1}b_s{}^{-}1 \cdots 1)$ is even, a contradiction as before. Thus, $b_1 = e_k$ and $b_j = 1$ for $1 \leqslant j \leqslant s$ and so $s = i - 1$. Thus, $[e_i \cdots e_{k-1}e_k1 \underline{\cdots 1}]/(e_i \cdots e_{k-1}e_k1 \cdots 1)$.

(iii) e_i represents ones.

Let $G = e_i \cdots e_{k-1}b_1 \cdots b_s$ be the first element in LC$(n-1)$ which begins with the string $e_i \cdots e_{k-1}$. Then certainly $b_1 \geqslant e_k$. Let $G' = a_1 \cdots a_t$ with $fG' = G$, and choose q such that $a_1 = e_i$, $a_2 = e_{i+1}$,..., $a_{q-1} = e_{i+q-2}$, and $a_q > e_{i+q-1}$. Then $q \leqslant k - i$, since G was the first lexicographic composition which began with the string $e_i \cdots e_{k-1}$. And $a_j = 1$ for $q < j \leqslant t$, otherwise we could find a lexicographic composition between G and G'. Now $t - q \geqslant i - 2$, since

$$n - 1 = \sum_{j=1}^{q} a_j + t - q = \sum_{j=i}^{k-1} e_j + 1 + t - q$$

$$\leqslant \sum_{j=i}^{k} e_j + t - q = (n-1) - (i-2) + t - q.$$

Thus, $\{a_1 \cdots a_q 1 \underline{\cdots 1}\}/[e_i \cdots e_{k-1}b_1 \cdots b_s]$.

As in the proof of Case 4, if a periodic composition arises, we apply Lemma 5 and proceed *mutatis mutandis*.

THEOREM 5. *The sequence generated by Algorithm* B *is a deBruijn sequence.*

Proof. Theorem 4 shows that every binary n-tuple appears at least once in the sequence generated by Algorithm B. Thus there must be at least 2^n bits in the sequence. But by Corollary 2, we have at most 2^n bits in the sequence. So we have precisely 2^n bits and the sequence is deBruijn.

4. The Sequence of Eldert et al.

The sequence of this paper and the sequence of Eldert are identical for $n \leqslant 6$. For $n \leqslant 14$ and $n = 18$, they differ only when our sequence uses compositions whose parts are exclusively two's and one's. Our conjecture is that the first nonlexicographic composition used in the Eldert sequence is the composition $2, 2, \ldots, 2, 1 \cdots 1, 2$, where the string of ones is t long, and $4 - t$ is the residue of $n - 1$ modulo 4. For example, in the case $n = 7$, the sequence of Eldert is

1111111000000011111011110011110100000100001100001011100011001000111011011001100101101
1010/0110101000/10100100101010.

The bits between the slashes are the bits where our sequence differs from the Eldert sequence in the case $n = 7$.

References

1. C. Eldert, H. J. Gray, H. M. Gurk, and M. Rubinoff, Shifting counters, *AIEE Trans.* **77** (1958), 70–74.
2. H. Fredricksen, The lexicographically least deBruijn cycle, *J. Combinatorial Theory* **9** (1970), 1–5.
3. H. Fredricksen, Generator of the Ford sequence of length 2^n, n large, *J. Combinatorial Theory* **12** (1972), 153–154.
4. H. Fredricksen, A class of non-linear deBruijn cycles, *J. Combinatorial Theory* **19** (1975), 192–199.
5. H. Fredricksen, On a deBruijn cycle algorithm, IDA-CRD Working Paper No. 431.
6. H. Fredricksen and I. J. Kessler, An algorithm for generating necklaces of beads in two colors, (to be published).
7. S. W. Golomb, "Shift Register Sequences," Holden–Day, San Francisco, 1967.
8. M. H. Martin, A problem in arrangements, *Bull. Amer. Math. Soc.* **40** (1940), 859–864.
9. T. Motzkin, Ordered and cyclic partitions, *Riveon Lematematica* **1** (1957), 61–67
10. J. Riordan, "An Introduction to Combinatorial Analysis," Wiley, New York, 1958